Part 1: (Not the correct answers!!)

The basic principle of color mixing is that you add the X components together, add the Y components together and add the Z components together. Start by making vectors containing the three tristimulus values for colors 1 and 2. Since the color matching space is entirely linear, you can mix two colors simply by adding their X, Y, and Z components to give the new color. R makes this addition simple: if you have a vector of three numbers, and add it to another vector of three numbers, the result is a vector which is the sum of all the first elements, all the second elements and all the third elements. Here is an example.

```
# Make vectors for each color
c1 <- c(X = 1, Y = 2, Z = 3)
c2 <- c(X = 5, Y = 7, Z = 11)
# add color c1 and c2
c3 <- c1 + c2
c1

## X Y Z
## 1 2 3

```

```
c2

## X Y Z
## 5 7 11

```

```
c3

## X Y Z
## 6 9 14

```

Note that c3 is the sum of the components of c1 and c2. That is 1 + 5 = 6. The same with components 2 (e.g. c1[2] and c2[3]) and components 3.

**Chromaticity**

Since the chromaticity (x, y, z) are the proportion of X, Y, and Z, in order to compute chromaticity all you need to do is divide each tristimulus value (X, Y, or Z) by the total amount of light (X + Y + Z) to get the chromaticity values.

```
# compute chromaticity
chrom1 <- c1 / sum(c1)
chrom2 <- c2 / sum(c2)
chrom3 <- c3 / sum(c3)
```
Graphs (these graphs do not need to be modified to produce plots)

Define the function for making the chromaticity plots. The mixture of colors \( c_1 + c_2 \) should lie on the straight line connecting \( c_1 \) and \( c_2 \).

# plot Figure 1
plot1 <- function()
{
  par(pty = "s")
  plot(0, 0, type = "n",
       xlim = c(0, 1),
       ylim = c(0, 1),
       xlab = "C.I.E. chromaticity x",
       ylab = "C.I.E. chromaticity y",
       main = "C.I.E. Chromaticity Diagram")
  abline(c(1,-1), col = "gray")
  points(chrom1[1], chrom1[2], pch = 19)
  text(chrom1[1], chrom1[2], "c1", pos = 4, cex = 0.7)
  points(chrom2[1], chrom2[2], pch = 19)
  text(chrom2[1], chrom2[2], "c2", pos = 4, cex = 0.7)
  points(chrom3[1], chrom3[2], pch = 19)
  text(chrom3[1], chrom3[2], "c1+c2", pos = 4, cex = 0.7)
  lines(c(chrom1[1], c(chrom2[1])), c(chrom1[2], chrom2[2]))
  par(pty = "m")
}
plot1()
Part 2 (Not the correct answers)

The basic principle is that the L-cone activity is multiplied by its coefficient, the M-cone by its coefficient and the S-cone by its coefficient and the three numbers are then added together to give a single number. It is helpful, but not necessary, to define the response properties of the three chromatic mechanism as functions, using the coefficients provided in the homework assignment. I am including the luminance channel here even though it is not needed for the homework. But if you had to compute many points in the red-green, yellow blue color space, having functions means that you would have to do a lot less coding! In the code below the input vector (three values M, L, S cone activity) is multiplied (dot product or scalar product) with the three values of the weights for each cone type. In R the scalar or inner product is carried out by the %*% operator that you see in the return() statement of each function.

```r
# define functions for each of the three chromatic mechanisms
rg <- function(cones) {
  # cones is a vector containing L, M, S cone activities
  # weights for the red-green mechanism
  wgt.rg <- c(1, -5, 0.1)
  return(cones %*% wgt.rg)
}

yb <- function(cones) {
  # computes the linear yellow-blue mechanism response
  # weights for the yellow-blue linear mechanism
  wgt.yb <- c(0.2, -0.2, -1)
  return(cones %*% wgt.yb)
}

lu <- function(cones) {
  # computes luminance from L, M, S cone activity
  wgt.lu <- c(1, 1, 1)
  return(cones %*% wgt.lu)
}

Define the two colors by their cone absorption amounts

```r
# define the colors from quantal absorption
c4 <- c(L = 10, M = 15, S = 8)
c5 <- c(L = 20, M = 5, S = 2)
c6 <- c4 + c5

Plot colors 4 and 5, plus the mixture, color 6, in the red-green, yellow-blue color space. Note that, unlike in the C.I.E. chromaticity space, mixtures of colors do not necessarily lie on a straight line between the two component colors. The plots below use the rg() and yb() functions defined above to compute the response of the channels.

```r
# plot Figure 2
plot2 <- function(){
  par(pty = "s")
  plot(0, 0, type = "n",
       xlim = c(-50, 50),
       ylim = c(-50, 50),
       xlab = "- (green) + (red)",
       ylab = "- (blue) + (yellow)",
       main = "Opponent-Process Color Space")
  abline(v=0, col = "gray")
}
```
abline(h=0, col = "gray")

# plot the color points
points(rg(c4), yb(c4), pch = 19)
text(rg(c4), yb(c4), "c4", pos = 4, cex = 0.7)

points(rg(c5), yb(c5), pch = 19)
text(rg(c5), yb(c5), "c5", pos = 4, cex = 0.7)

points(rg(c6), yb(c6), pch = 19)
text(rg(c6), yb(c6), "c4+c5", pos = 4, cex = 0.7)
par(pty = "m")
}
plot2()

Opponent–Process Color Space

Make another figure with both graphs in it.

plot.both <- function(){
  par(mfcol=c(1,2))  # two graphs per figure
  plot1()
  plot2()
  par(mfcol=c(1,1))  # one graph per figure
}
plot.both()
C.I.E. Chromaticity Diagram

C.I.E. chromaticity x

C.I.E. chromaticity y

Opponent–Process Color Space

− (green)                  + (red)

− (blue)                   + (yellow)

c1 + c2

c4 + c5

C.I.E. Chromaticity Diagram

C.I.E. Chromaticity x

Opponent–Process Color Space

− (green)                  + (red)